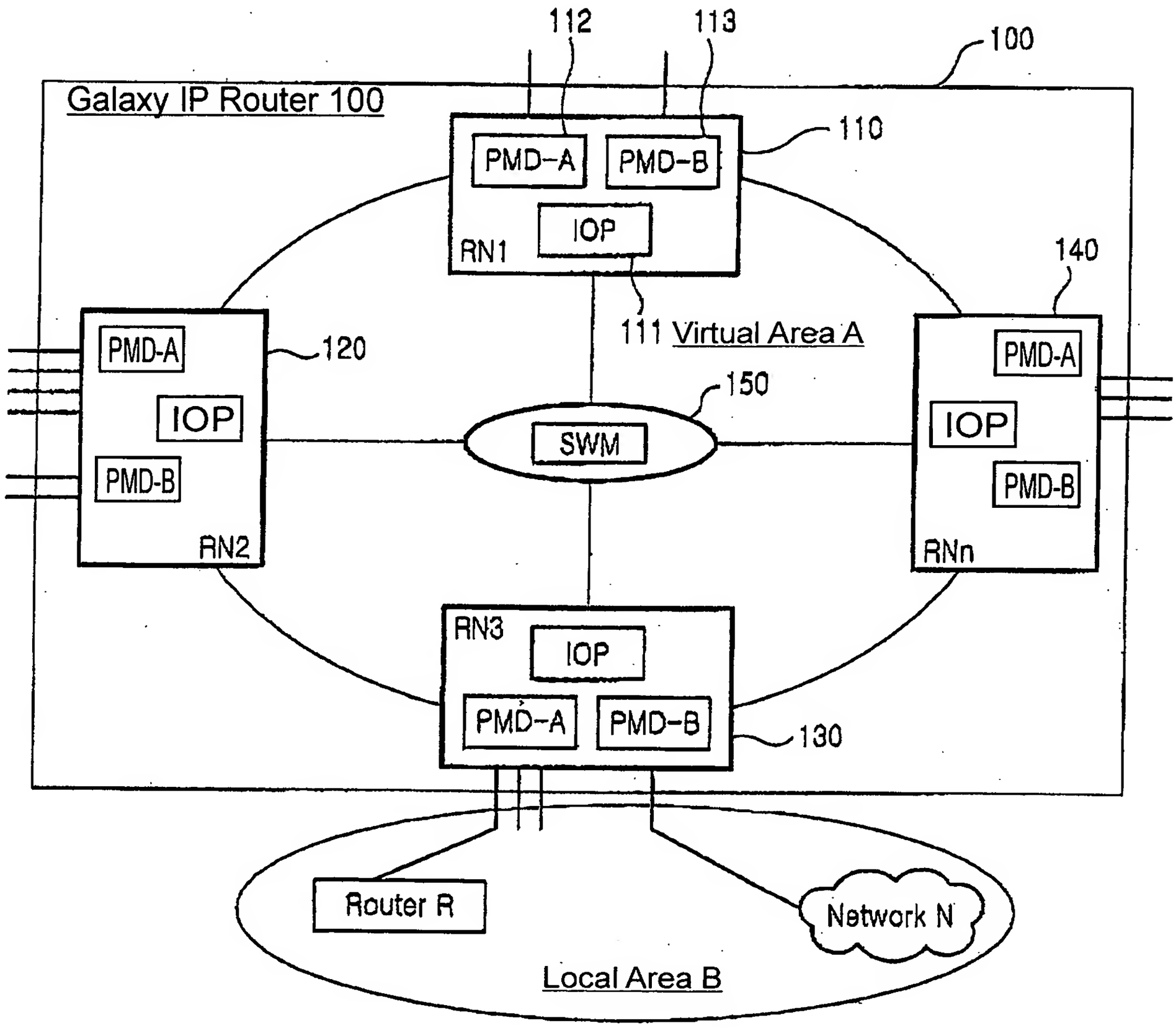

112
113
100
Galaxy IP Router 100
PMD-A
PMD-B
110
IOP
RN1
140
111 Virtual Area A
PMD-A
120
150
IOP
SWM
PMD-B
PMD-A
IOP
PMD-B
RN2
RNn
RN3
IOP
PMD-A
PMD-B
130
Router R
Network N
Local Area B

FIG.1

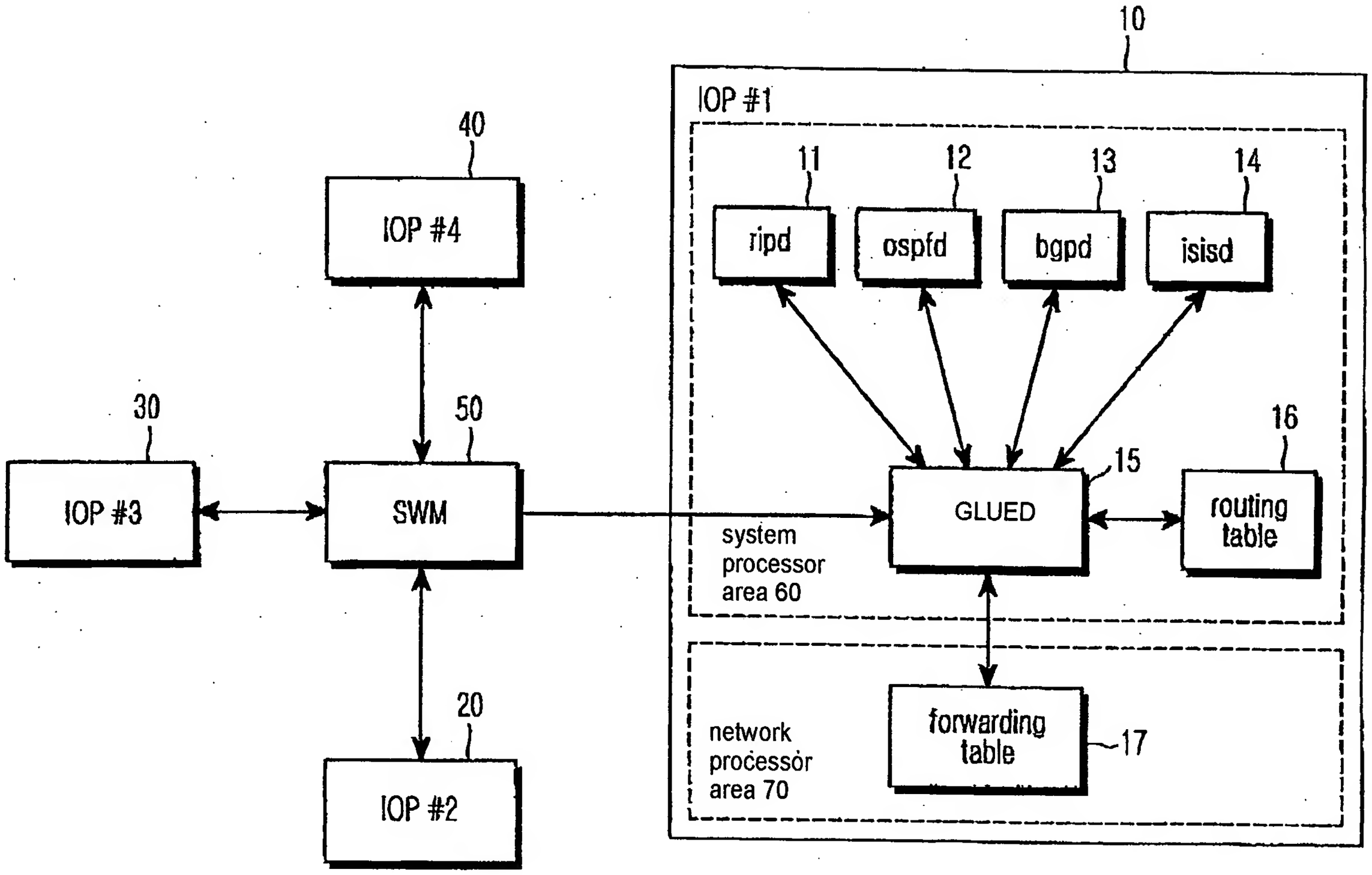

10
IOP #1
11
12
13
14
ripd
ospfd
bgpd
isisd
40
IOP #4
30
50
16
15
IOP #3
SWM
GLUED
routing table
system processor area 60
network processor area 70
forwarding table
17
20
IOP #2

FIG.2

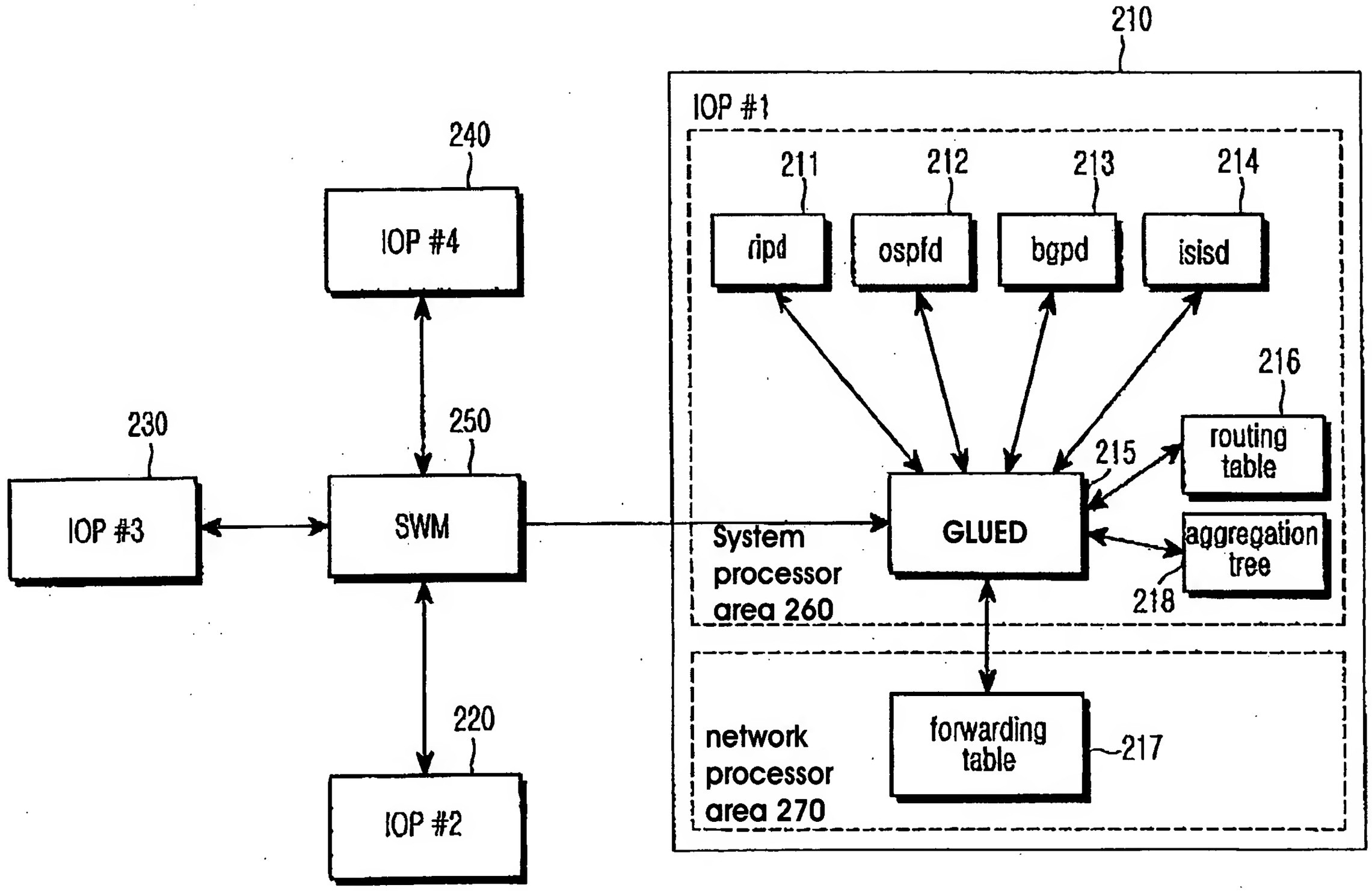

210
IOP #1
240
IOP #4
211
212
213
214
ripd
ospfd
bgpd
isisd
216
routing table
230
250
215
IOP #3
SWM
GLUED
System processor area 260
aggregation tree
218
220
network processor area 270
forwarding table
217
IOP #2

FIG.3

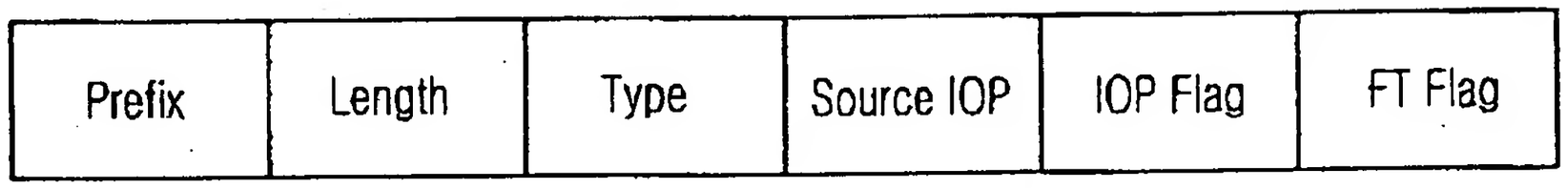

Prefix
Length
Type
Source IOP
IOP Flag
FT Flag

FIG.4A

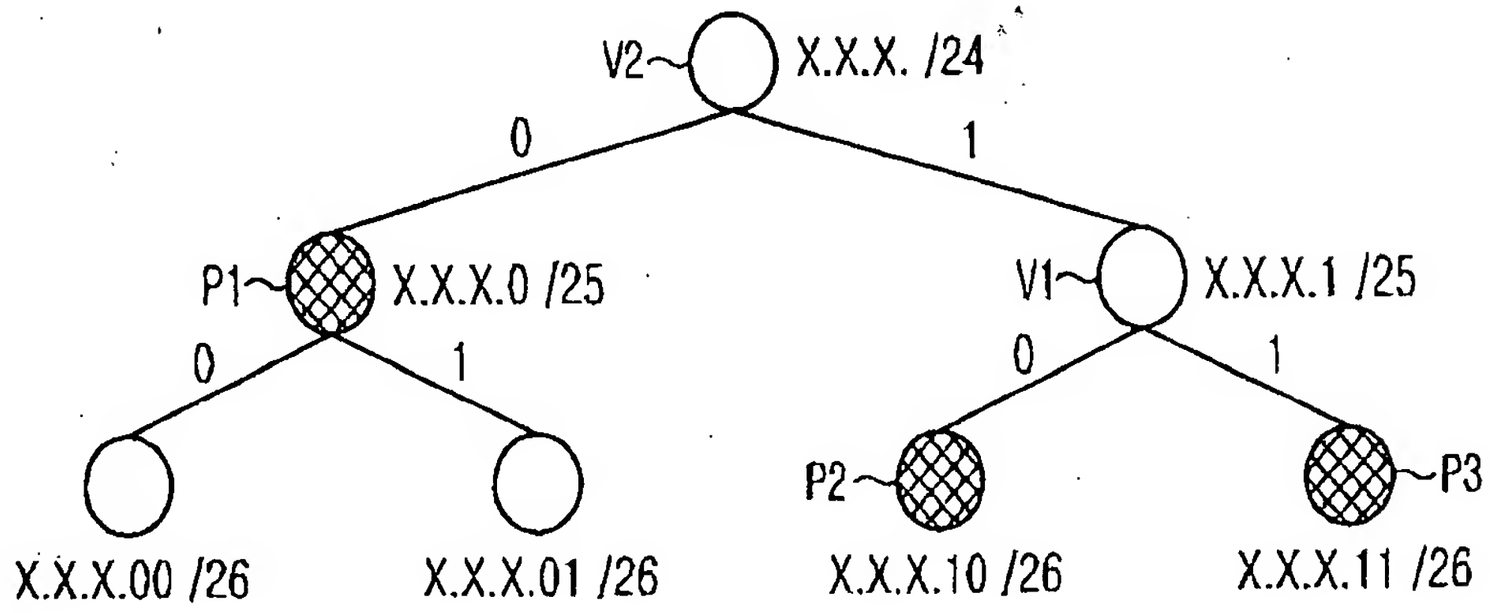

V2
X.X.X. /24
0
1
P1
X.X.X.0 /25
V1
X.X.X.1 /25
0
1
0
1
P2
P3
X.X.X.00 /26
X.X.X.01 /26
X.X.X.10 /26
X.X.X.11 /26

FIG.4B

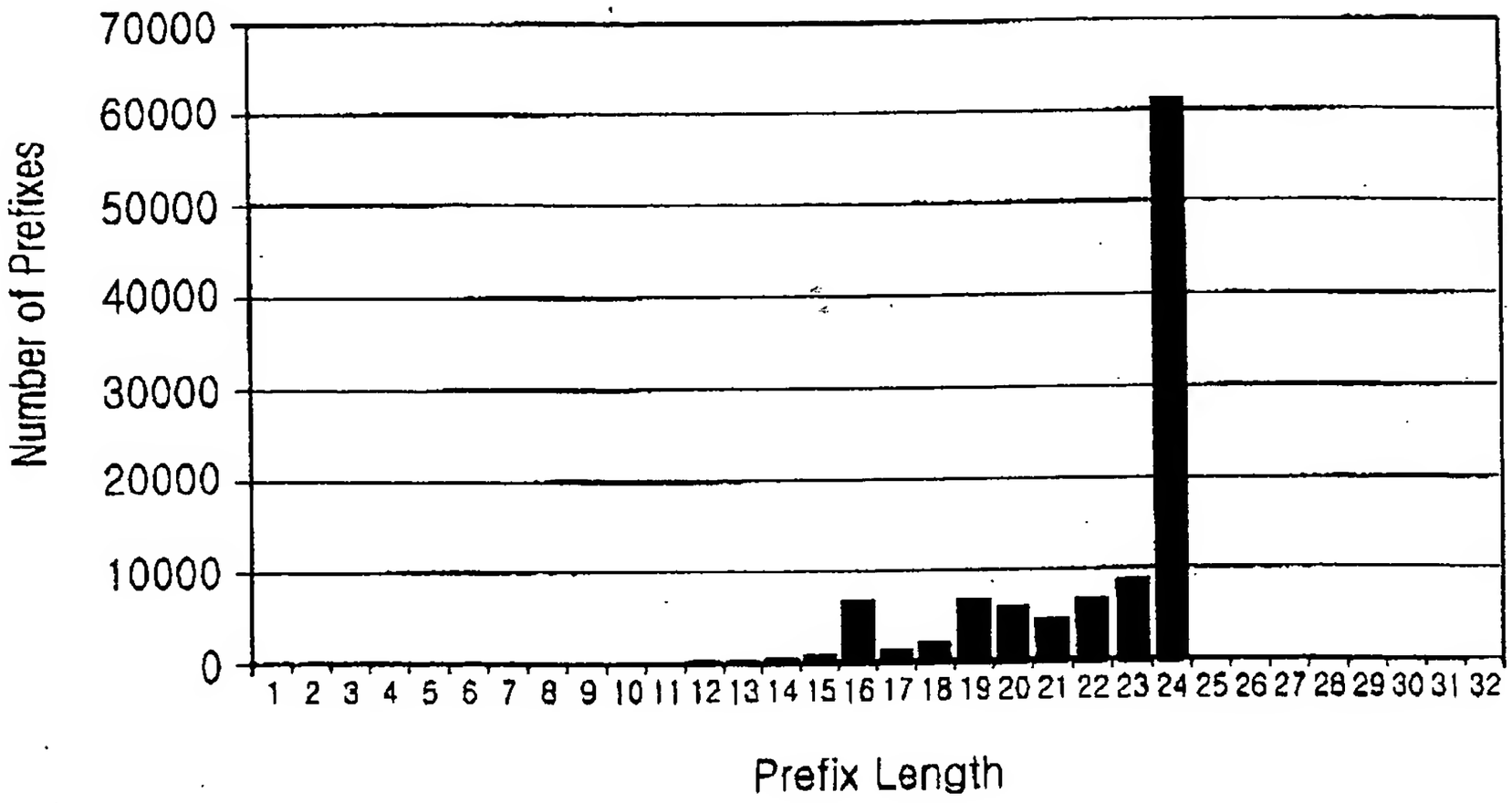

Number of Prefixes
70000
60000
50000
40000
30000
20000
10000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Prefix Length

FIG.4C

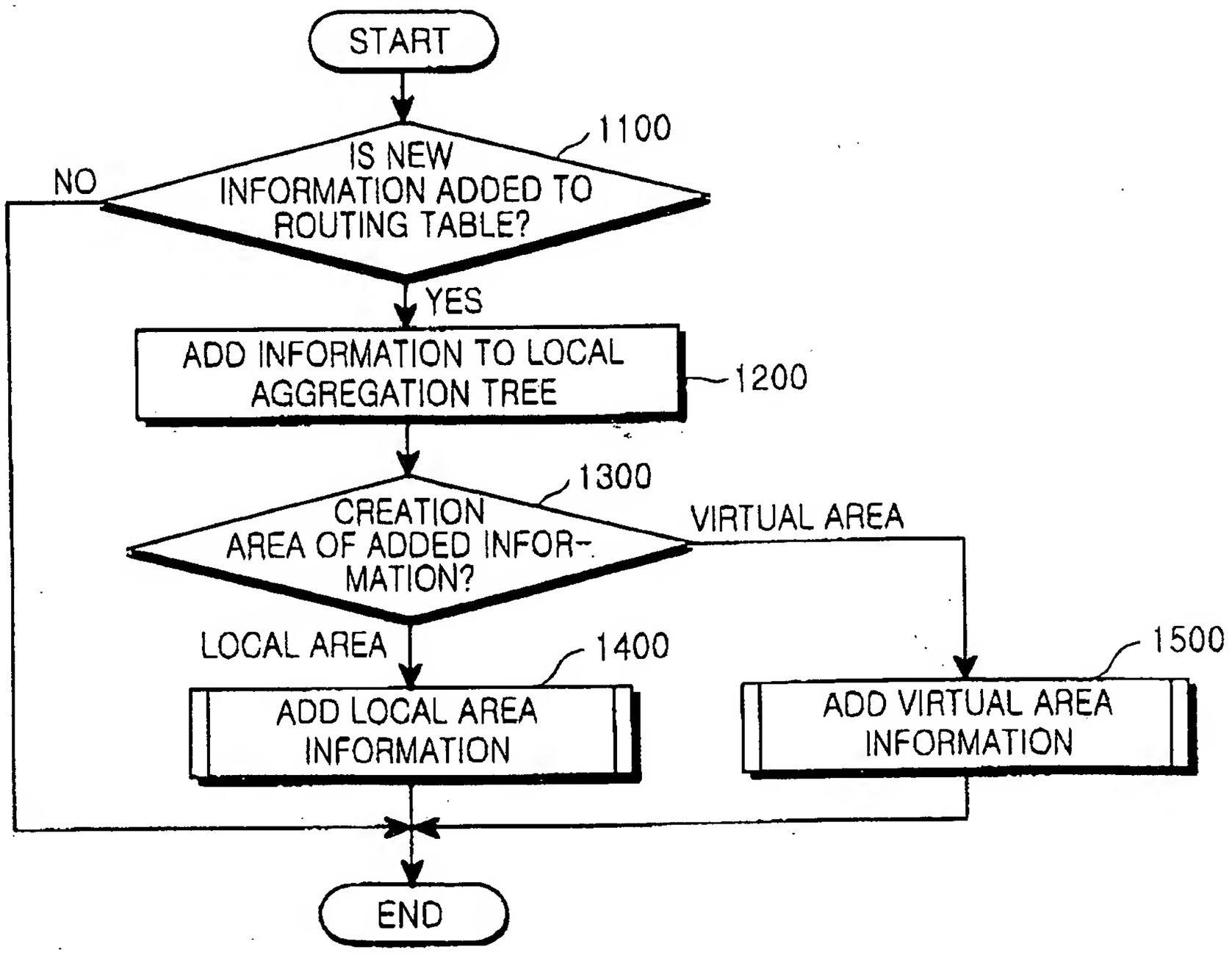

START
IS NEW INFORMATION ADDED TO ROUTING TABLE?
1100
NO
YES
ADD INFORMATION TO LOCAL AGGREGATION TREE
1200
CREATION AREA OF ADDED INFORMATION?
1300
VIRTUAL AREA
LOCAL AREA
ADD LOCAL AREA INFORMATION
1400
ADD VIRTUAL AREA INFORMATION
1500
END

FIG.5

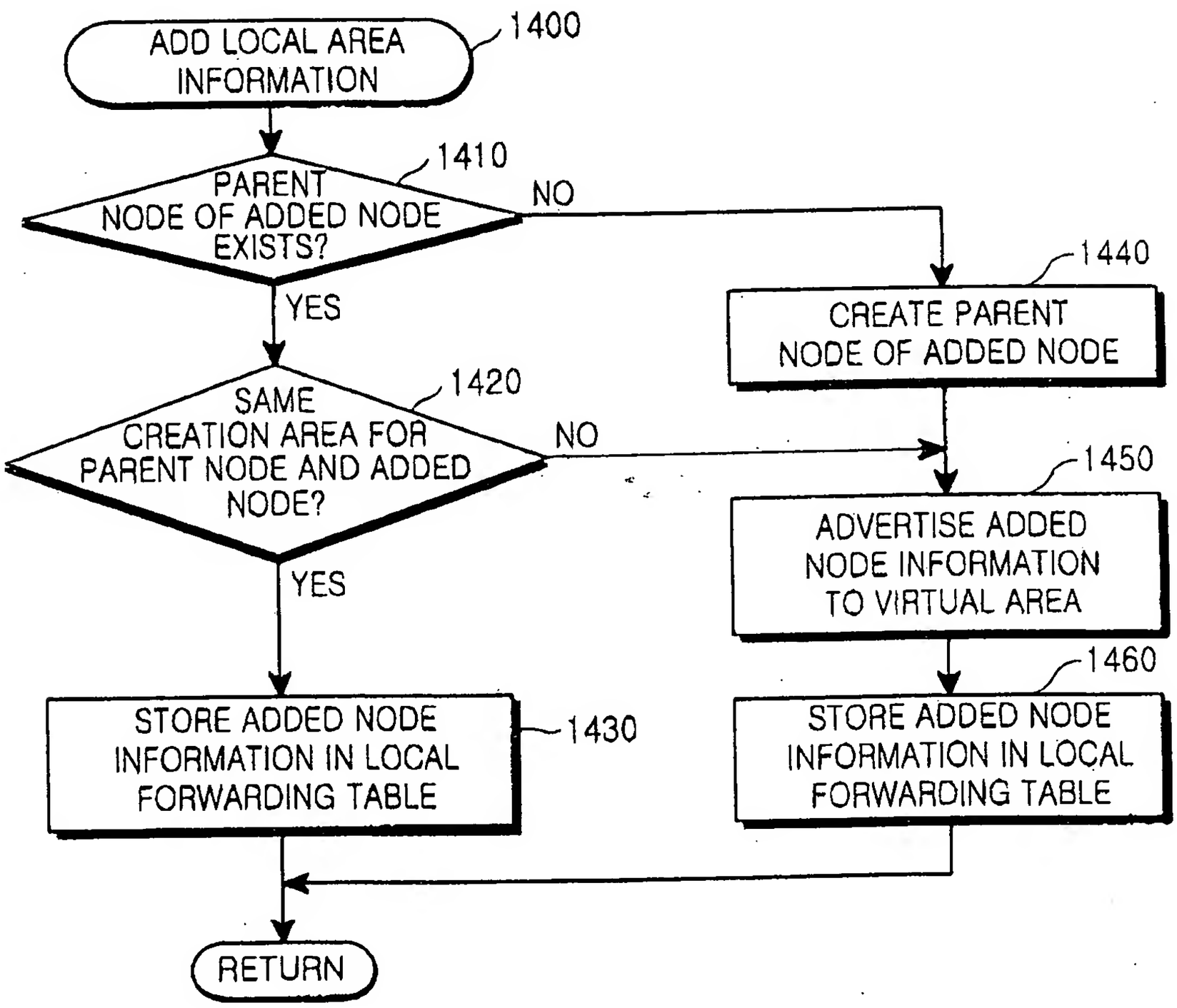

ADD LOCAL AREA INFORMATION
1400
PARENT NODE OF ADDED NODE EXISTS?
1410
NO
YES
1440
CREATE PARENT NODE OF ADDED NODE
SAME CREATION AREA FOR PARENT NODE AND ADDED NODE?
1420
NO
YES
1450
ADVERTISE ADDED NODE INFORMATION TO VIRTUAL AREA
1460
STORE ADDED NODE INFORMATION IN LOCAL FORWARDING TABLE
1430
STORE ADDED NODE INFORMATION IN LOCAL FORWARDING TABLE
RETURN

FIG.6

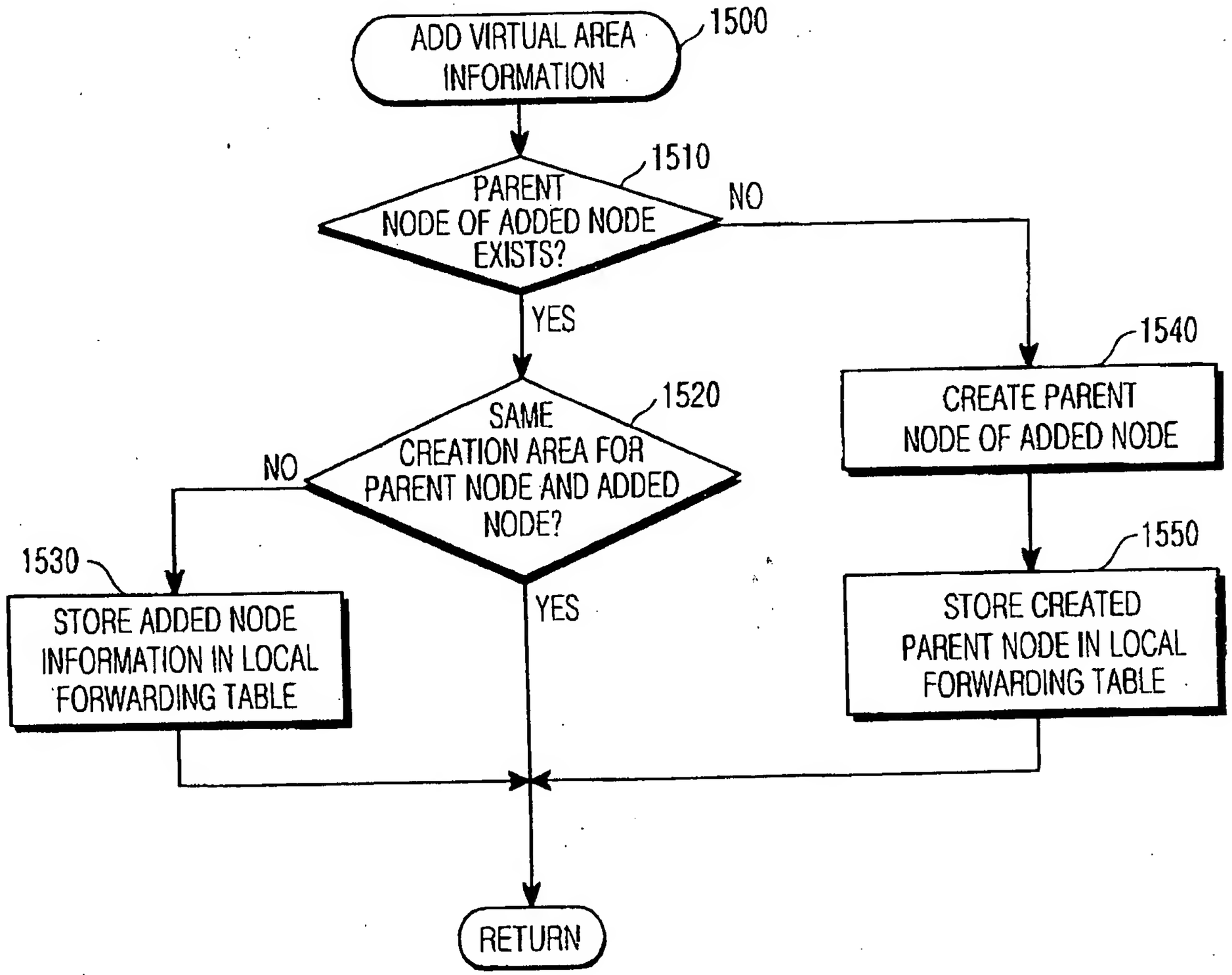

ADD VIRTUAL AREA INFORMATION
1500
PARENT NODE OF ADDED NODE EXISTS?
1510
NO
YES
SAME CREATION AREA FOR PARENT NODE AND ADDED NODE?
1520
NO
YES
CREATE PARENT NODE OF ADDED NODE
1540
1530
STORE ADDED NODE INFORMATION IN LOCAL FORWARDING TABLE
STORE CREATED PARENT NODE IN LOCAL FORWARDING TABLE
1550
RETURN

FIG.7

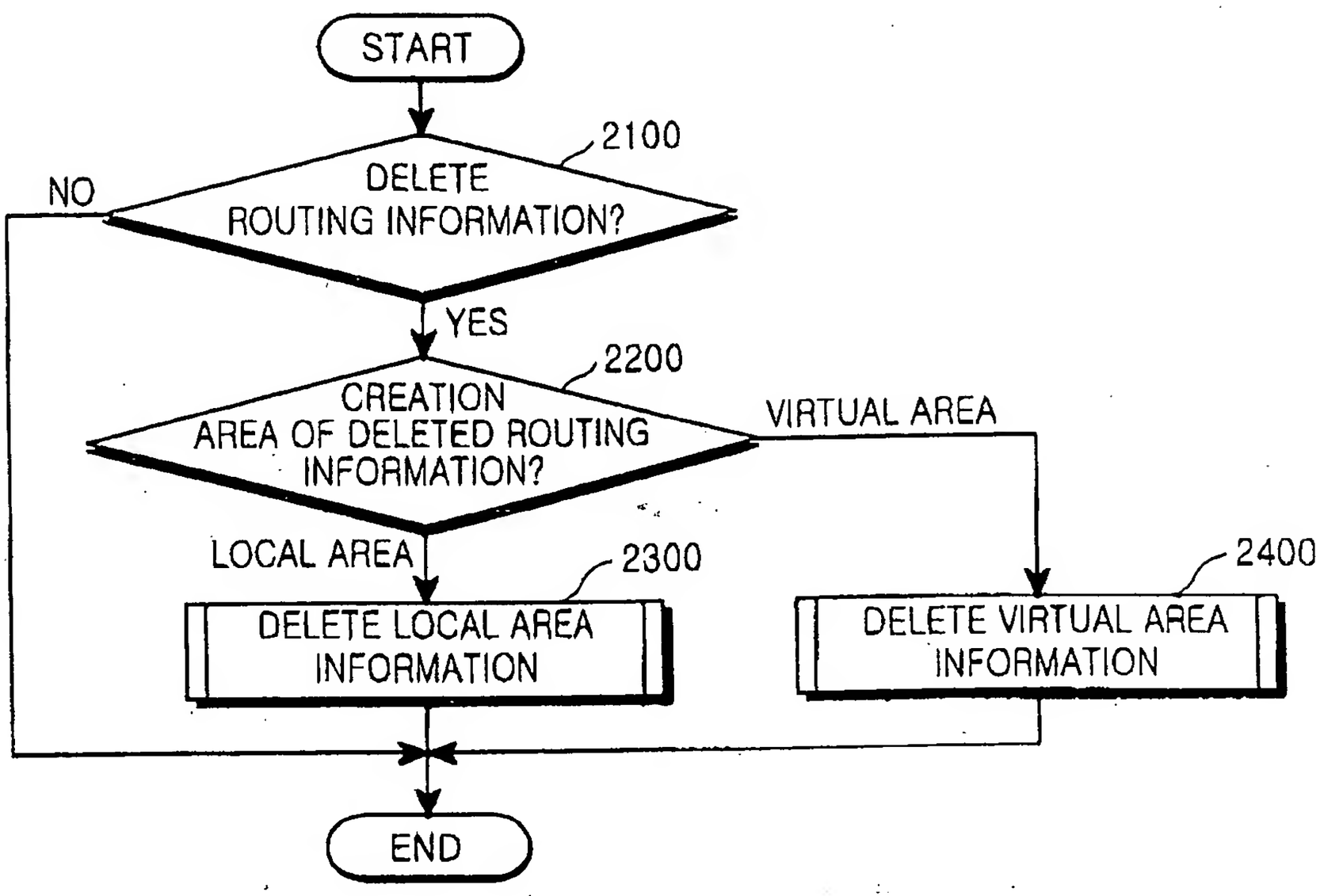

START
2100
DELETE ROUTING INFORMATION?
NO
YES
2200
CREATION AREA OF DELETED ROUTING INFORMATION?
VIRTUAL AREA
LOCAL AREA
2300
DELETE LOCAL AREA INFORMATION
2400
DELETE VIRTUAL AREA INFORMATION
END

FIG.8

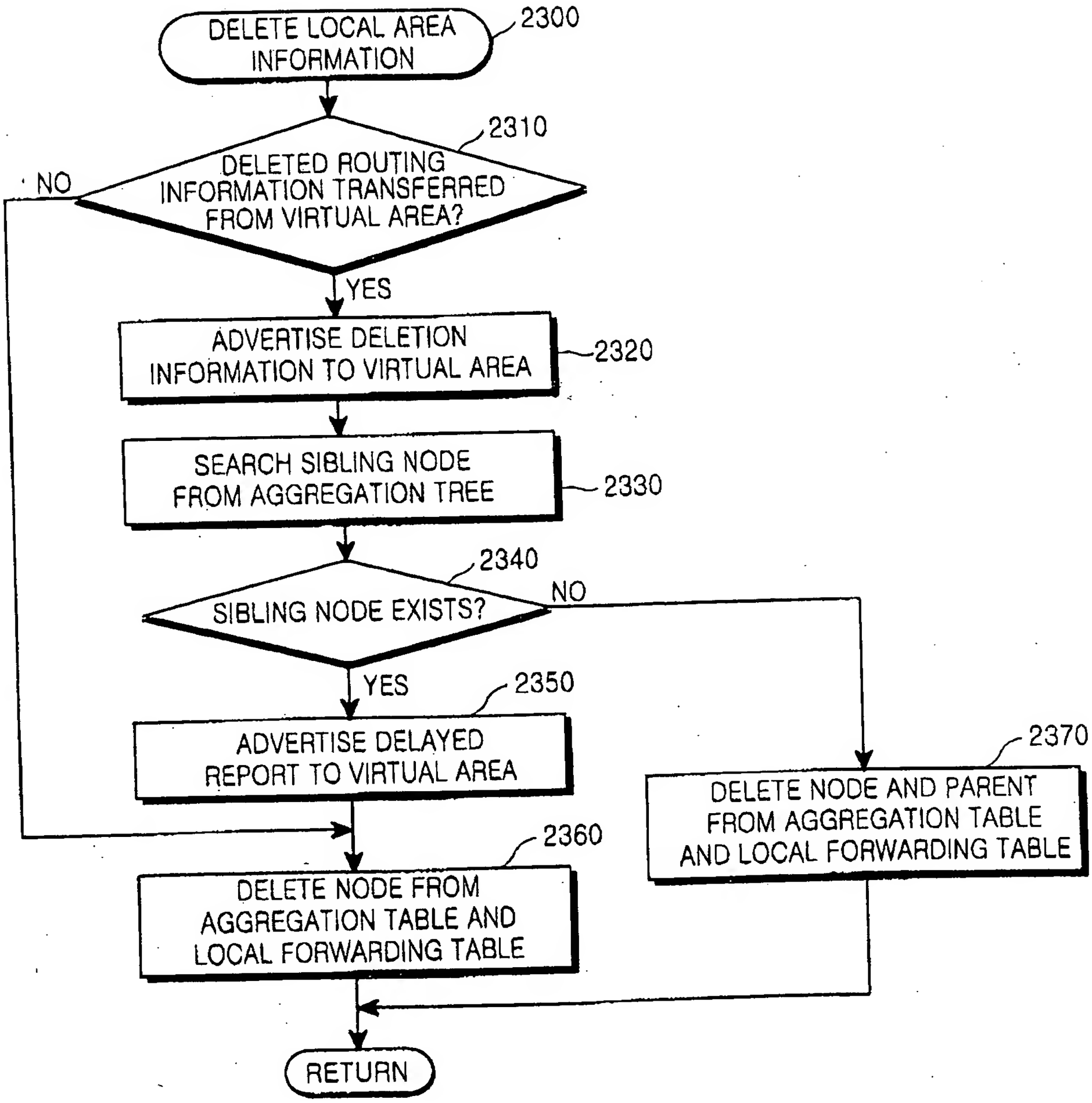

DELETE LOCAL AREA INFORMATION
2300
DELETED ROUTING INFORMATION TRANSFERRED FROM VIRTUAL AREA?
2310
NO
YES
ADVERTISE DELETION INFORMATION TO VIRTUAL AREA
2320
SEARCH SIBLING NODE FROM AGGREGATION TREE
2330
SIBLING NODE EXISTS?
2340
NO
YES
ADVERTISE DELAYED REPORT TO VIRTUAL AREA
2350
DELETE NODE AND PARENT FROM AGGREGATION TABLE AND LOCAL FORWARDING TABLE
2370
DELETE NODE FROM AGGREGATION TABLE AND LOCAL FORWARDING TABLE
2360
RETURN

FIG.9

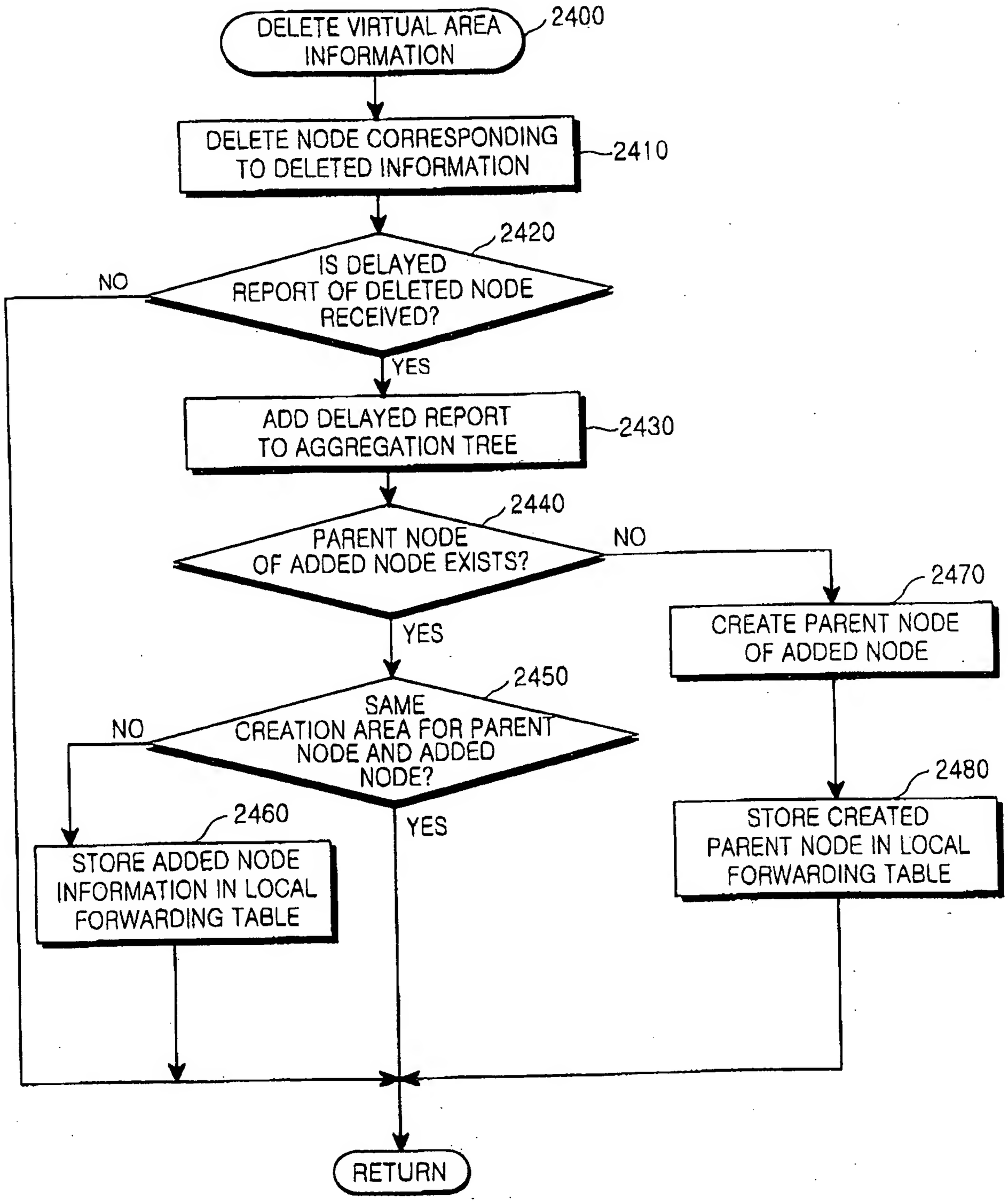

DELETE VIRTUAL AREA INFORMATION
2400
DELETE NODE CORRESPONDING TO DELETED INFORMATION
2410
2420
IS DELAYED REPORT OF DELETED NODE RECEIVED?
NO
YES
ADD DELAYED REPORT TO AGGREGATION TREE
2430
2440
PARENT NODE OF ADDED NODE EXISTS?
NO
YES
2470
CREATE PARENT NODE OF ADDED NODE
2450
SAME CREATION AREA FOR PARENT NODE AND ADDED NODE?
NO
YES
2480
STORE CREATED PARENT NODE IN LOCAL FORWARDING TABLE
2460
STORE ADDED NODE INFORMATION IN LOCAL FORWARDING TABLE
RETURN

FIG.10

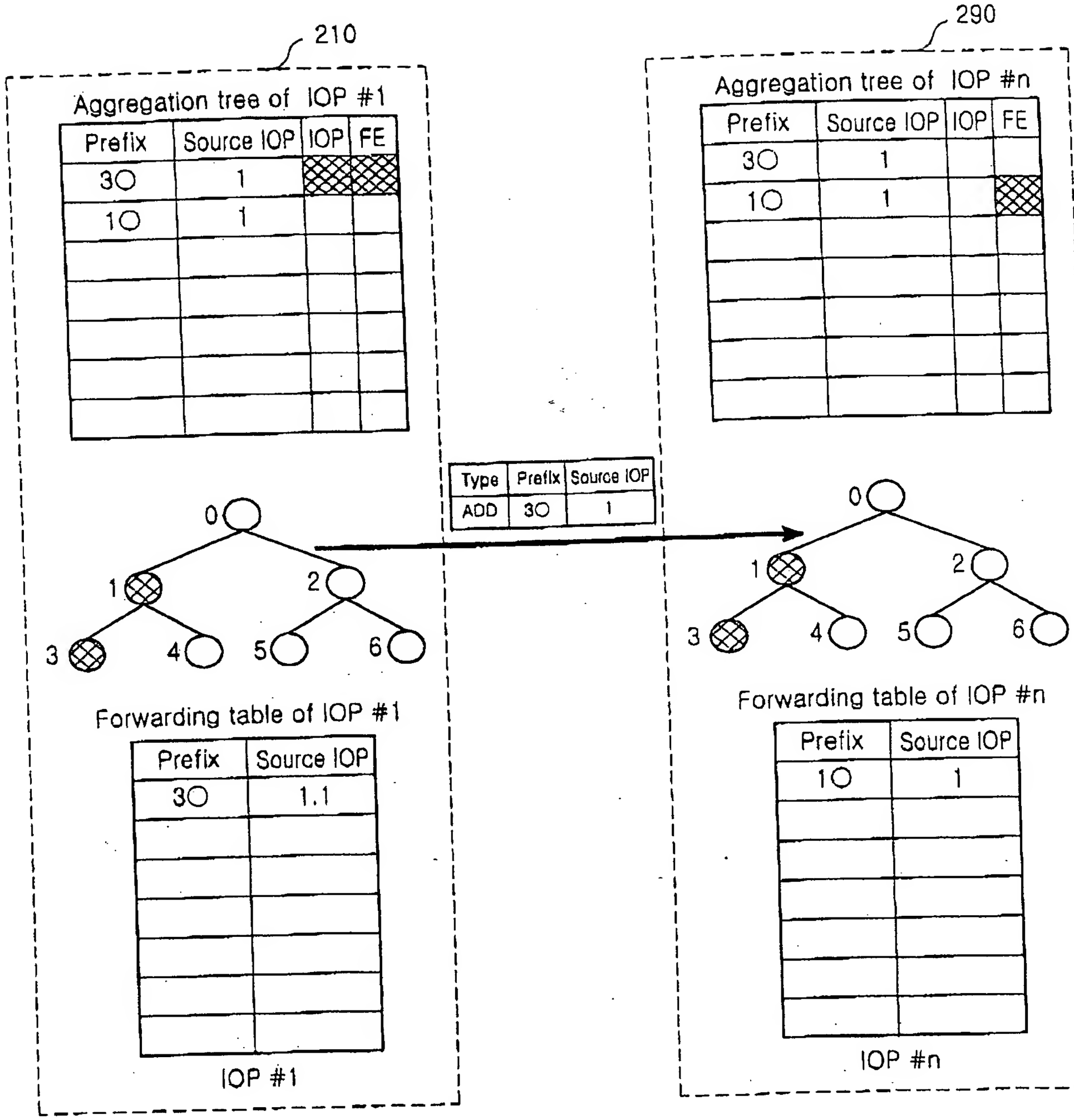

210
Aggregation tree of IOP #1
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
Forwarding table of IOP #1
Prefix
Source IOP
3O
1.1
IOP #1
Type
Prefix
Source IOP
ADD
3O
1
290
Aggregation tree of IOP #n
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
Forwarding table of IOP #n
Prefix
Source IOP
1O
1
IOP #n

FIG.11A

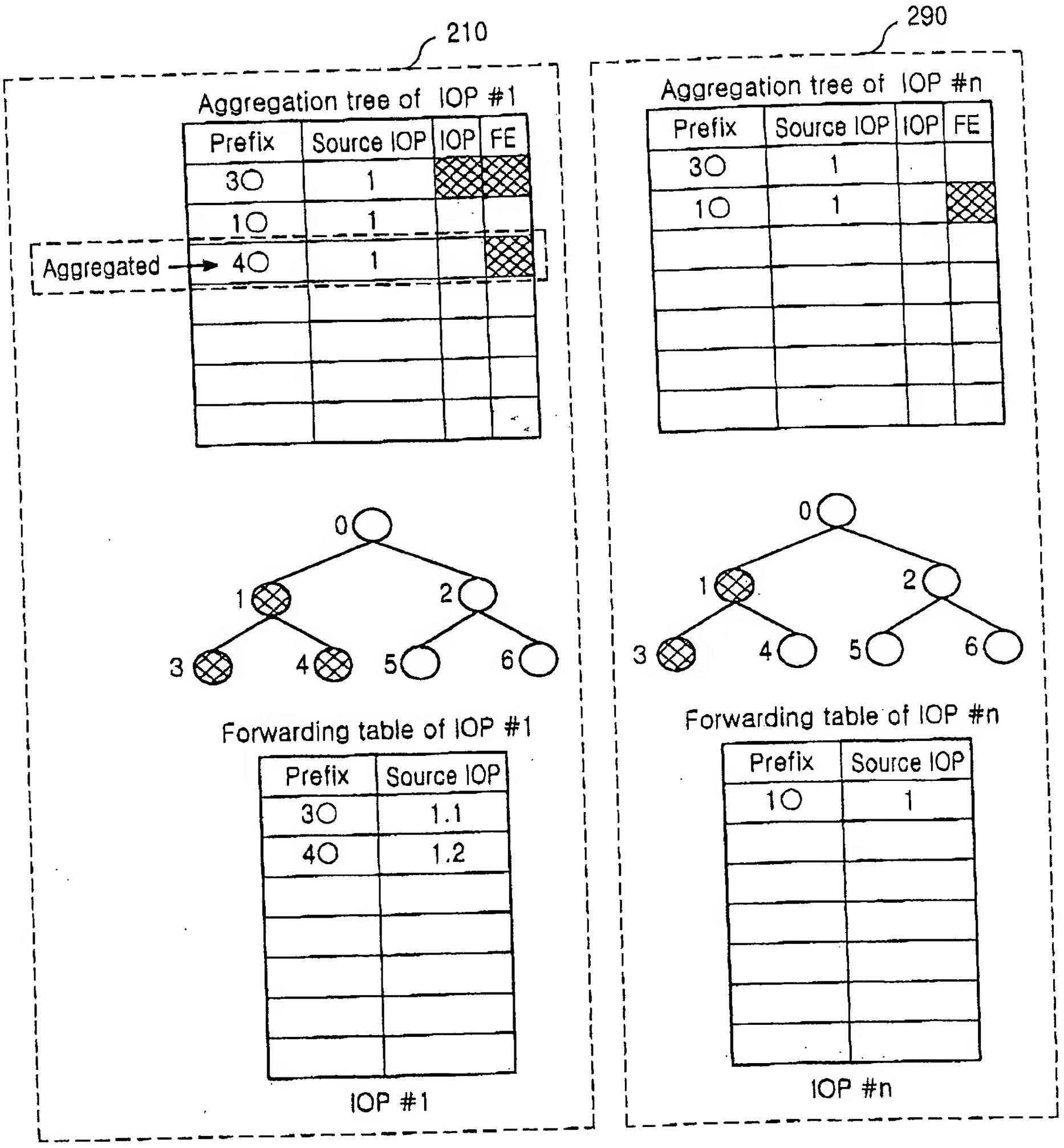

210
Aggregation tree of IOP #1
Prefix
Source IOP
IOP
FE
30
1
10
1
Aggregated
40
1
0
1
2
3
4
5
6
Forwarding table of IOP #1
Prefix
Source IOP
30
1.1
40
1.2
IOP #1
290
Aggregation tree of IOP #n
Prefix
Source IOP
IOP
FE
30
1
10
1
0
1
2
3
4
5
6
Forwarding table of IOP #n
Prefix
Source IOP
10
1
IOP #n

FIG.11B

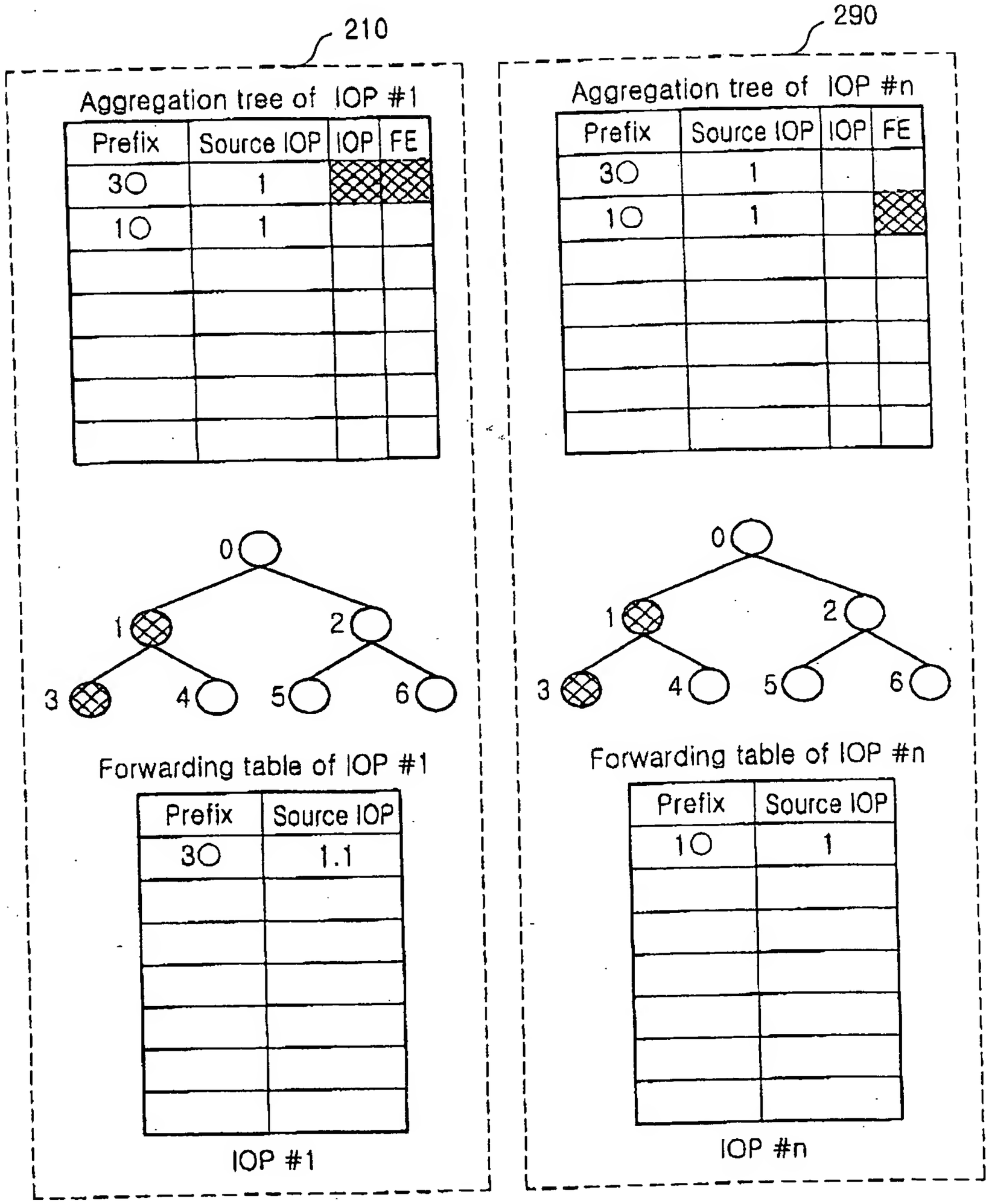

210
Aggregation tree of IOP #1
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
Forwarding table of IOP #1
Prefix
Source IOP
3O
1.1
IOP #1
290
Aggregation tree of IOP #n
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
Forwarding table of IOP #n
Prefix
Source IOP
1O
1
IOP #n

FIG.12A

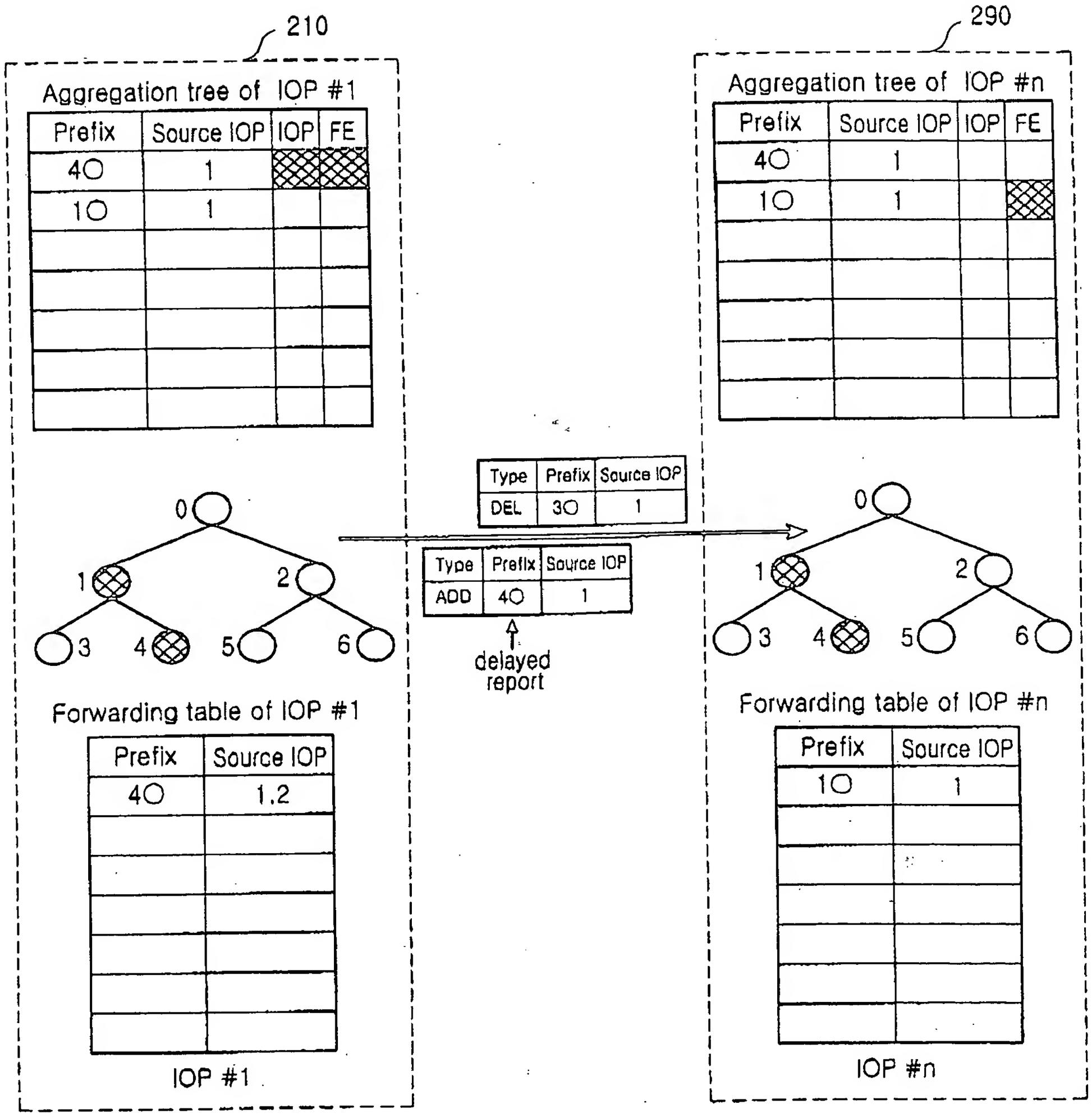
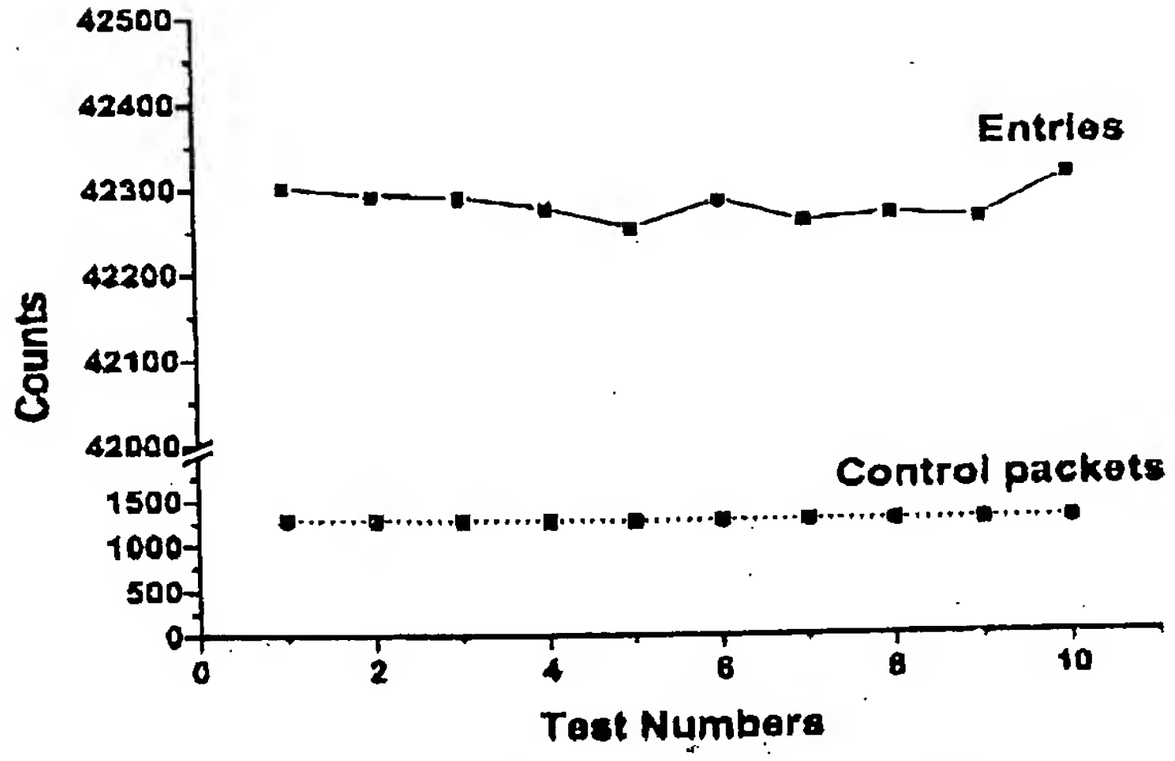
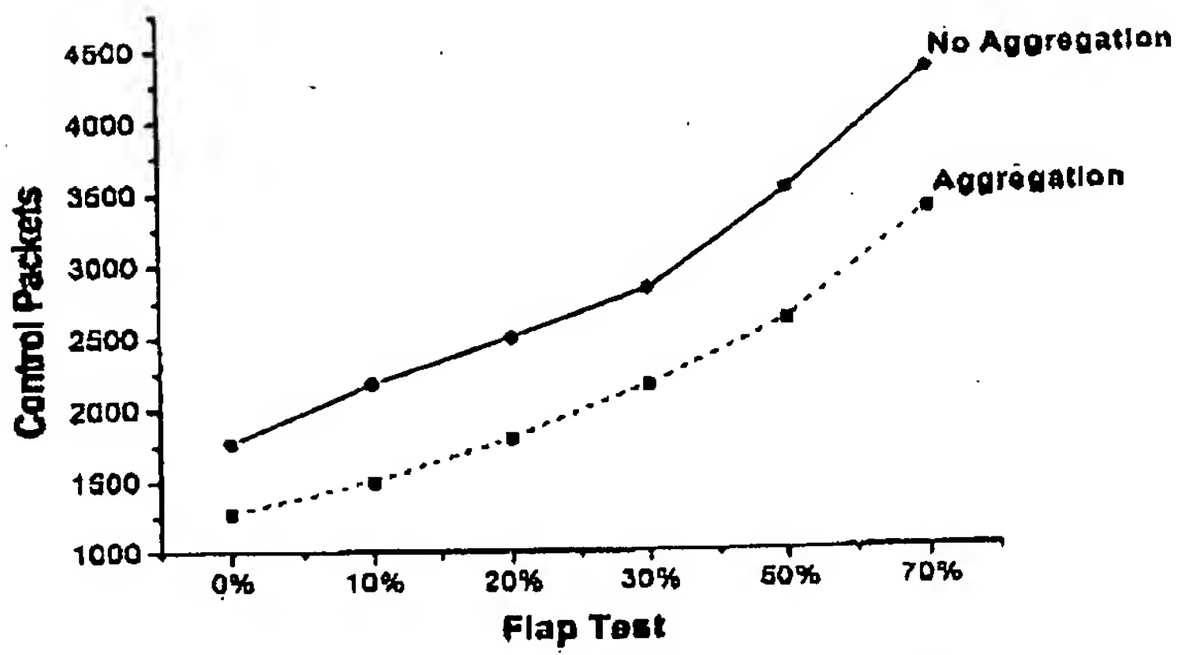

210
Aggregation tree of IOP #1
Prefix
Source IOP
IOP
FE
40
1
10
1
0
1
2
3
4
5
6
Forwarding table of IOP #1
Prefix
Source IOP
40
1.2
IOP #1
Type
Prefix
Source IOP
DEL
30
1
Type
Prefix
Source IOP
ADD
40
1
delayed report
290
Aggregation tree of IOP #n
Prefix
Source IOP
IOP
FE
40
1
10
1
0
1
2
3
4
5
6
Forwarding table of IOP #n
Prefix
Source IOP
10
1
IOP #n

FIG.12B



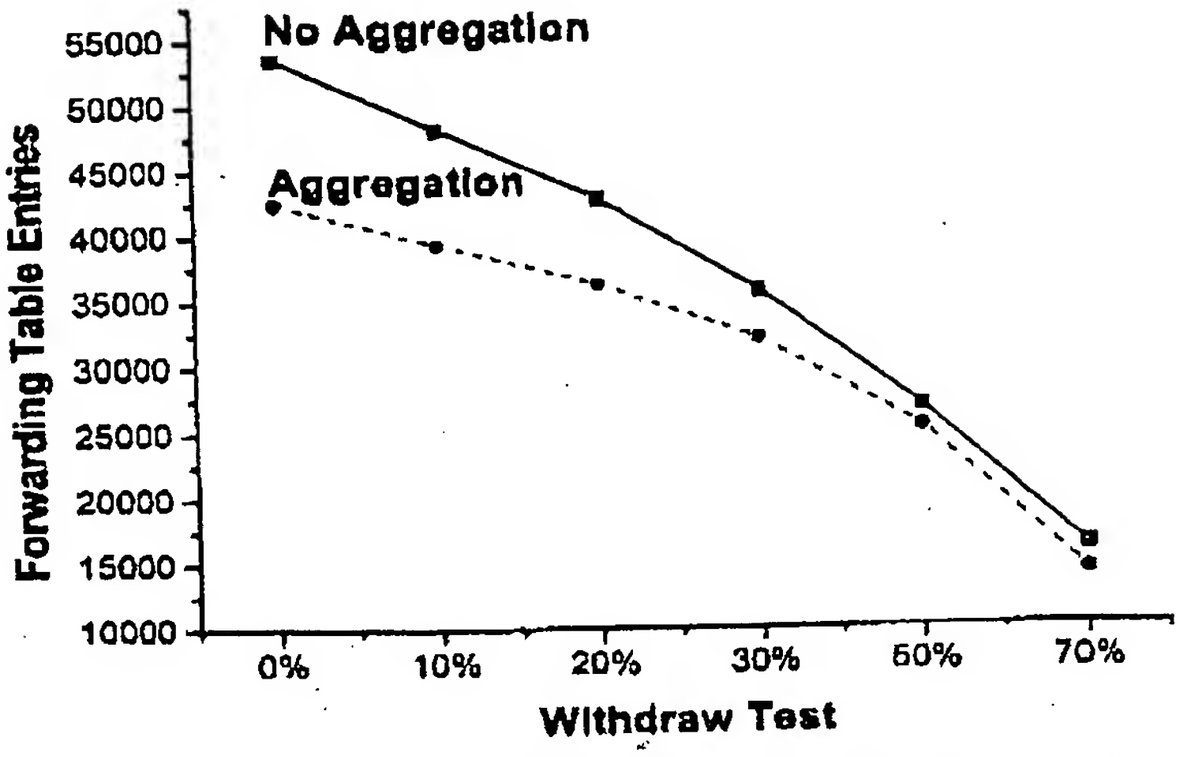
RANDOM ORDER ENTRY TEST

FIG.13A



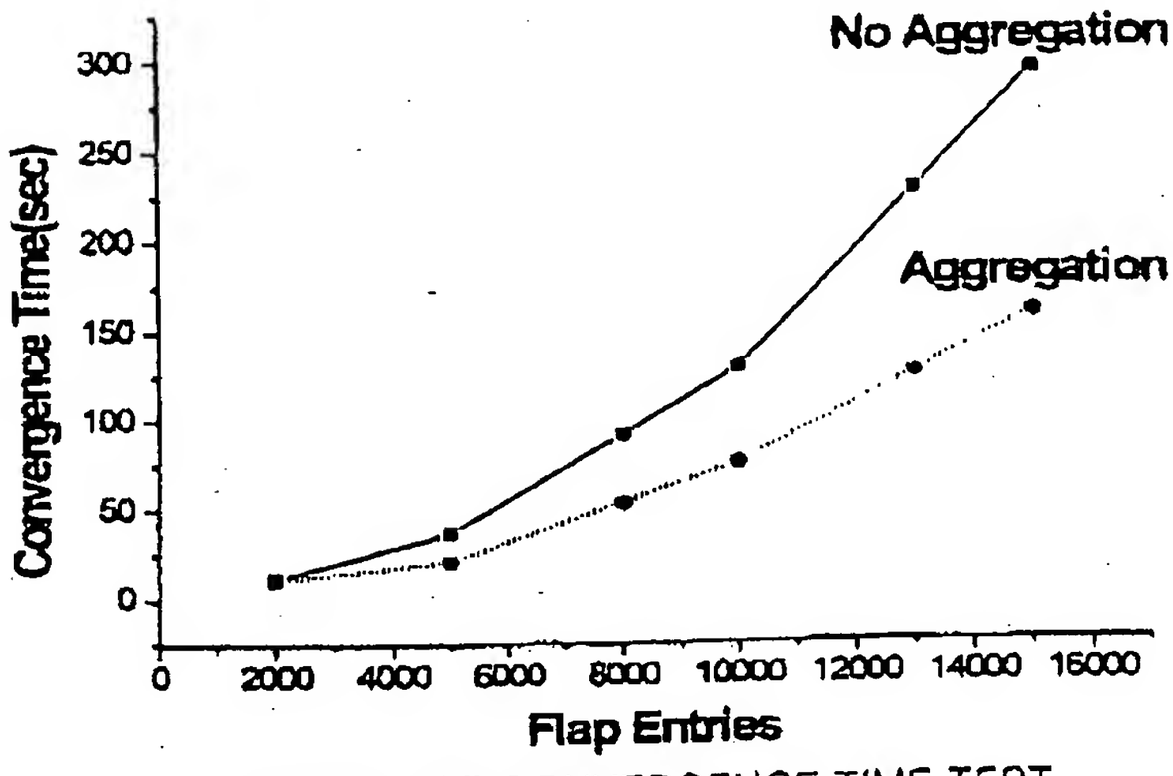
CONTROL PACKET TRANSMISSION

FIG.13B



NUMBER OF ENTRIES IN FORWARDING TABLE

FIG.13C



BGP4 FLAP CONVERGENCE TIME TEST

FIG.13D

```
Insertion (prefix) {
 local insertionnode
 /* STEP1 */
 insertionnode := FindNode(prefix)
 /* STEP2 */
 /* Check new route's nexthop */
 if (NodeNexthopVirtual(insertionnode) = TRUE) then
    InsertionNodeVirtual(insertionnode)
 else
    InsertionNodeInter-domain(insertionnode)
 /* STEP3 */
 if (run_step3 = TRUE) then
    ChildNodeHandler(insertionnode)
}
```

FIG.14A

```
Deletion (prefix) {
  local deletenode, siblingnode
  /* STEP1 */
  deletenode := FindNode(prefix)
  /* STEP2 */
  if (DRNForward(deletenode) = TRUE) then
    if (ForwardingTableForward(deletenode) = TRUE)
      then
        DeleteForwardingTable(deletenode)
        SendDRN(deletenode)
      else
        /* Deletion of the parent node instead of  */
        /* the deletenode                          */
        DeleteForwardingTable(GetParent(deletenode))
  else
     /* In case of the deletion of the aggregation node, */
     /* sending information to DRN can be suppressed */
    if (ForwardingTableForward(deletenode) = TRUE)
      then
        DeleteForwardingTable(deletenode)

  siblingnode := SiblingNodeCheck(deletenode)
  / * Delayed insertion of the sibling node */
  if (siblingnode ≠ NIL) then
     SendDRN(siblingnode)
  /* STEP3 */
  ChildNodeHandler (deletenode)
}
```

FIG.14B

```
FindNode (prefix) {
   local node
   /* Search for node to be inserted */
   node := GetNode (prefix)
   /* Identity the insertion node */
   if ((node ? NIL) and (NodeType(node) = AGG)) then
         Empty (node)
         run_step3 := TRUE
   else
         NewNode(node)
   return (node)
}
```

FIG.14C

```
MakeParentNode (node) {
 local parentnode
 /* Make parent node and set node type to AGG */
 parentnode := AllocateParentNode(node)
 NodeType(parentnode) := AGG
 Parent(node) := parentnode
 return(parentnode)
}
```

FIG.14D

```
InsertNodeVirtul (node) {
 local parentnode
 parentnode := GetParent(node)
 if (parentnode ? NIL) then
  if (NodeSource(parentnode) ? NodeSource(node)) then
          InsertForwardingTable(node)
  /* There is not parent node. */
 else
  /* Make and wirte parent node to the forwarding table */
     parentnode := MakeParentNode(node)
     InsertForwardingTable(parentnode)
}
```

FIG.14E

```
InsertNodeInter-domain (node) {
 local parentnode
 parentnode := GetParent(node)
 if (parentnode ? NIL) then
     /* If new route source and parent are same, */
     /* new route sending to DRN can be suppressed */
  if (NodeSource(parentnode) ? NodeSource(node)) then
          SendDRN(node)
          InsertForwardingTable(node)
  else
     /* Make parent node and wirte new node */
/* to the forwarding table */
     parentnode := MakeParentNode(node)
     InsertForwardingTable(node)
     SendDRN(node)
 }
```

FIG.14F

```
ForwardingTableForwardCheck(node)
{
   local parentnode
   parentnode := GetParent(node)
   if ((ForwardingTableForward(node) = FALSE)
   and (NodeSource(node) != NodeSource(parentnode))) then
         return (FALSE)
   else
         return (TRUE)
}
```

FIG.14G

```
DRNForwardCheck(node)
{
   local parentnode
   parentnode := GetParent(node)
   if ((DRNForward(node) = FALSE)
      and (NodeNexthopVirtual(node) = FALSE)) then
         return (FALSE)
   else
         return (TRUE)
}
```

FIG.14H

```
ChildNodeHandler (node) {
  local leftchild, rightchild
  leftchild := GetLeftChildNode(node)
  rightchild := GetRightChildNode(node)
  if (leftchild ? NIL) then
     /* Disaggregation of the leftchild node */
   if ((ForwardingTableForwardCheck(leftchild) = FALSE)  then
            InsertForwardingTable(leftchild)
      /* Delayed insertion of the leftchild node */
      if ((DRNForwardCheck(leftchild) = FALSE) then
            SendDRN(leftchild)
   if (rightchild ? NIL) then
      /* Disaggregation of the rightchild node */
   if ((ForwardingTableForwardCheck(rightchild) = FALSE) then
            InsertForwardingTable(rightchild)
       /* Delayed insertion of the rightchild node */
      if ((DRNForwardCheck(rightchild) = FALSE) then
            SendDRN(rightchild)
}
```

FIG.14I

```
SiblingNodeCheck (node) {
   local siblingnode
   siblingnode := GetSiblingNode(node)
   /* Check if there is aggregated sibling node */
   if ((NodeSource(node) = NodeSource(siblingnode))
        and (DRNForward(siblingnode) = FALSE)) then
         return (siblingnode)
   else
         return (FALSE)
}
```

FIG.14J